Nama : Muhammad Riza Kumalasari

NIM : 23104410112

Matkul : Sistem Operasi

TUGAS 02

Jawab pertanyaan berikut

1. Mengapa pada System Calls lebih sering menggunakan API daripada Direct system call?
2. Jika pada Linux terdapat Kernel Linux, maka pada IOS terdapat Darwin Kernel. Gambarkan artirekturnya, jelaskan beberapa bagian pentingnya dan apa hubungannya dengan Mach Microkenrnel?

Jawab:

1. Pemakaian API memberikan *portabilitas* (Kemampuan kode untuk berjalan di berbagai platform tanpa banyak perubahan), *stabilitas* antarpublikasi, penanganan kesalahan dan *errno* (Variabel yang menyimpan kode kesalahan dari fungsi sistem/ pustaka setelah terjadi kegagalan); perilaku tambahan (buffering, cancellation, emulasi), dan sering kali performansi/optimasi yang lebih baik dibandingkan memanggil syscall (*system calls*) secara langsung.

   Memanggil syscall langsung berguna untuk hal-hal khusus (*low-level, runtime language* tertentu, atau ketika ingin menghindari *libc*), tapi untuk aplikasi umum API lebih aman dan mudah dipelihara.

2. Darwin adalah *base Unix-like* OS Apple. Di pusatnya ada XNU (*X is Not Unix*) sebuah hybrid kernel yang menggabungkan konsep *Mach microkernel* (untuk IPC, task/thread, memory objects) dan BSD (untuk *POSIX, filesystem, network*, proses), plus IOKit (driver model berbasis C++). XNU bukan murni microkernel: ia mengintegrasikan banyak layanan di kernel untuk alasan performa → disebut hybrid.

   Gambaran sederhana:

   Userland (apps, frameworks: UIKit, Foundation, libSystem)
   → **System Libraries** (libSystem, libc+libm+libdispatch, security frameworks)
   → **Darwin Kernel (XNU)** :
   • Mach layer (primitive: tasks, threads, ports, IPC, VM primitives)
   • BSD layer (POSIX APIs, process model, VFS, sockets, userland syscalls)
   • I/O Kit (driver framework berbasis C++/IOService)
   • Kernel extensions / DriverKit (model driver modern)
   → **Hardware** (ARM64/x86_64)
   (Skema ini sesuai penjelasan arsitektural Darwin/XNU).

Bagian-bagian penting dan perannya :

a. **Mach (inti microkernel yang dimodifikasi)**

   Tanggung jawab utama: objek-objek dasar OS (task, thread), manajemen virtual memory low-level, IPC berbasis ports (pesan), dan trap/exception handling. XNU memakai Mach sebagai basis tetapi telah dimodifikasi (OSF MK 7.3 turunan Mach 3.x) untuk kebutuhan Apple. Mach memberi abstraksi untuk isolasi dan messaging antar-komponen.

b. **BSD layer (FreeBSD / BSD-derived subsystem)**

   Menyediakan POSIX API: proses, user/group, standar syscalls (open/read/write/fork/exec), VFS (virtual filesystem), networking stack (BSD sockets), signals, dan utilitas lain yang membuat Darwin bersifat Unix-like. BSD juga membawa banyak kode berfungsi untuk performance dan kompatibilitas POSIX. BSD berjalan “di atas” Mach primitives dalam XNU — memberikan antarmuka yang familier untuk aplikasi.

c. **IOKit (driver framework C++ khusus Apple)**

   Struktur object-oriented untuk driver (IOService, IORegistry). IOKit memungkinkan driver berjalan di kernel namun ditulis dalam subset C++ sehingga lebih modular. DriverKit dan substitusi KEXT modern (user-space drivers) muncul sebagai evolusi keamanan.

d. **Kernel extensions (KEXT) dan DriverKit**

   Mekanisme untuk menambah modul kernel (device drivers, file systems). Karena alasan keamanan, Apple bergerak ke model driver yang berjalan di user space (DriverKit) untuk mengurangi attack surface. KEXT lebih berisiko karena kode pihak ketiga berjalan di kernel.

e. **Hybrid design (alasan praktisnya)**

   Mach microkernel murni menjanjikan isolasi dan modularity lewat message passing, tapi biaya pesan/IPC dan overhead sering menurunkan performa. XNU mengadopsi Mach primitives tapi menggabungkan banyak layanan (BSD, networking, VFS) langsung di kernel untuk menghindari overhead pesan berulang dan meningkatkan throughput. Jadi XNU sering disebut hybrid kernel (mengambil keuntungan microkernel + monolithic).

Hubungan XNU / Darwin dengan Mach microkernel — inti perbedaannya

- **Asal usul**: XNU dibangun dari Mach (CMU Mach) yang kemudian dimodifikasi (OSF MK turunan) dan disatukan dengan kode BSD/NeXTSTEP saat NeXT mengembangkannya; Apple melanjutkan pengembangan itu setelah mengakuisisi NeXT.

- **Bukan Mach murni**: XNU memakai Mach untuk primitives (task/thread, IPC, VM) tetapi tidak memaksakan bahwa semua layanan (filesystem, network) dijalankan sebagai server terpisah via IPC; banyak layanan dipasang langsung di kernel — ini penting untuk performa. Jadi: XNU = Mach primitives + BSD kernel services + IOKit, bukan microkernel klasikal yang memecah semuanya ke proses terpisah.

Contoh praktis bagaimana Mach memengaruhi Darwin

- **IPC & Ports:** macOS/iOS masih menggunakan konsep Mach ports untuk beberapa mekanisme pesan (mis. communication dengan launchd, Mach exceptions).
- **Virtual memory:** Mach mendesain model memory object yang memengaruhi bagaimana XNU menangani VM, copy-on-write, dan paging.
- **Kernel traps/exceptions:** mekanisme exception handling di XNU diwarisi dari Mach.

Contoh analogi sederhana

Bayangkan OS sebagai rumah sakit:
- ✓ Mach adalah struktur dasar gedung, lift, dan sistem pesan internal (bagaimana staf mengirim informasi antar ruangan).
- ✓ BSD adalah prosedur medis dan staf yang melakukan operasi sehari-hari (proses, jaringan, file).
- ✓ IOKit adalah peralatan medis khusus (driver perangkat).

  Darwin (XNU) adalah rumah sakit lengkap yang memakai fondasi Mach tetapi juga menempatkan banyak fungsi kritis (dokter/ruang operasi) di dalam gedung itu sendiri supaya penanganan pasien cepat — jadi bukan gedung yang hanya memfasilitasi pengiriman pesan antar-unit saja.

Intisari cepat dan takeaways

Gunakan API/library untuk kebanyakan aplikasi: lebih portable, lebih aman, lebih mudah dipelihara, dan seringkali lebih cepat secara praktis karena optimisasi (buffering, batching).

Darwin / iOS dibangun di atas XNU, sebuah hybrid kernel yang memadukan Mach (primitives) dengan BSD (POSIX, filesystem, network) dan IOKit (driver). XNU bukan microkernel murni — Apple mengorbankan sebagian isolasi microkernel demi performa dan kompatibilitas.